VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Bakalaurinis darbas

Identifikavimo schemos, naudojančios klaidas taisančius kodus
(Identification schemes using error-correcting codes)

Atliko: 4 kurso, 1 grupės studentas
Vytautas Stankus
Darbo vadovas:
Gintaras Skersys

Vilnius
2016

Turinys

Įvadas

Klaidas taisantys kodai daugiausia naudojamai aptikti ir ištaisyti klaidas, atsirandančias duomenų persiuntimo metu. Tačiau šie kodai gali būti pritaikyti ir kitose srityse tokiose kaip identifikavimo sistemos. Šio darbo tikslas yra ištirti tris identifikavimo schemas, naudojančias klaidas taisančius kodus: Stern, Harari ir Veron. Stern schema pasirinkta tyrimui nes ji yra viena iš dažniausia sutinkamu identifikavimo schemų, naudojančių klaidas taisančius kodus. Stern schema yra naudojama kaip lyginamasis taškas daugumai kitų schemų, todėl ji pasirinkta kaip tyrimo atsvaros taškas. Veron schema yra sukurta Stern schemos pagrindu siekiant ją patobulinti. Galiausia Harari schema pasirinkta šiam darbui, nes ji naudoja išskirtinę identifikavimo metodiką. Stern ir Veron schemos yra remtos trijų komunikacijų identifikacija, Harari schema – paraleliniu skaičiavimu.

Šiuo darbu siekiama: susipažinti su pagrindinėmis identifikavimo sąvokomis, susipažinti su pasirinktomis identifikavimo schemomis, ištirti bei palyginti teorinius jų įverčius bei jas interpretuojančias programas.

1. Pagrindinės sąvokos

1.1. Identifikavimas

Identifikavimas yra apibrėžiamas kaip procesas, kurio metu viena šalis, vadinama tikrinančiuoju, yra užtikrinama, kad kita šalis, vadinama pareiškėju, yra būtent tas, kuo skelbiasi esąs, ir kad ši šalis aktyviai esamu laiku dalyvauja procese.

Identifikavimas dažniausia yra naudojamas apriboti prieigą prie atitinkamų resursų, kai ta prieiga yra skirta tik atitinkamiems individams, pavyzdžiui prieiga prie atitinkamų kompiuterinių programų, pinigų išėmimas iš bankomatų, ar leidimas įeiti į saugomą pastatą. Paprasčiausią identifikavimą galime įsivaizduoti, kaip sąrašą ryšių, tarp individų ir resursų, kai sėkmingas tapatybės patvirtinimas garantuoja galimybę pasinaudoti resursu.

1.1.1. Identifikavimo tikslai

Paimkime paprastą pavyzdį, kur pareiškėjas bus A, o tikrinantysis B, taip pat egzistuoja trečiosios šalys C. Pagrindinis identifikavimo proceso tikslas yra, kad B būtų užtikrintas, kad A yra būtent tas A, kurio buvo tikėtasi. Tačiau identifikavimo procesas turi ir kitus tikslus:

- Normaliu atveju, kai nevyksta jokios apgavystės, A pavykus įrodyti B savo tapatybę, B priima prisijungimą.
- B negali pasinaudoti gautais duomenimis ir apsimesti, kad jis yra A trečiosioms šalims C.
- Šansas, kad trečioji šalis C apsimesdama A apgaus ir bus priimta B kaip A yra labai mažas.

- Ankstesni pareiškimai išlieka teisingi nesvarbu kiek identifikavimo procesų tarp A ir B galėjo stebėti trečiosios šalys C.[MOV96, §10.1.1]

1.1.2. Identifikavimo technikos

Paprastai identifikavimo technikos yra skirstomos į tris pagrindines grupes:

1. *Dalinis žinojimas:* tai atitinkamos informacijos žinojimas, kuri naudojama kaip identifikatorius. Šios technikos pavyzdžiai apima paprastus slaptažodžius, asmens identifikavimo numerius (PIN), bei privačius raktus gaunamus klausimų-atsakymų protokolais.
2. *Turimas objektas:* dažniausia fizinis objektas, funkcionuojantis kaip identifikatorius. Pavyzdžiai apima magnetines bei lustines korteles, kodų generatorius.
3. *Biometrika:* ši technologija naudoja žmogui unikalias savybes ar veiksmus, pavyzdžiui balsą, akių raineles, parašus, rankų geometriją ir t.t.

1.1.3. Identifikavimo protokolai

Identifikavimo protokolai gali turėti labai daug įvairių savybių, tačiau vartotojui šios savybės yra svarbiausios:

- *Identifikacijos abipusiškumas:* identifikacija gali būti vienpusė, arba abipusė. Abipusės identifikacijos atveju abi šalys bando įrodyti savo tapatybę viena kitai. Taikant kai kurias identifikavimo technikas, pavyzdžiui fiksuotų slaptažodžių schemas, identifikacijos abipusiškumas gali būti silpnoji vieta, kai trečios šalys gali bandyti apsimesti tikrinančiuoju, kad gauti pareiškėjo slaptažodį.
- *Skaičiavimų efektyvumas:* protokolo įvykdymui reikalingas operacijų kiekis.
- *Komunikacijos efektyvumas:* ši savybė apima informacijos apsikeitimų kiekį, bei pasikeičiamų paketų dydį

Sudėtingesnės identifikavimo protokolų savybės:

- *Trečiųjų šalių dalyvavimas:* ši savybė apima tiek patikimas trečiąsias šalis, teikiančias identifikavimo raktus, tiek nepatikimas trečiąsias šalis, teikiančias viešųjų raktų sertifikatus.
- *Trečiųjų šalių (jei tokių yra) patikimumo pobūdis:* pavyzdžiai apima patikėjimą trečiosioms šalims identifikuoti ir sujungti šalį su jos viešuoju raktu, ar pasitikėjimą atskleisti trečiosioms šalims kitos šalies privatųjį raktą.
- *Apsaugos garantijos pobūdis:* pavyzdžiui įrodoma apsauga ir informacijos neatskleidimo parametras.
- *Paslapčių kaupimas:* ši savybė apima kritinės raktų informacijos saugojimo vietą bei būdą. [MOV96, §10.1.2]

1.2. Informacijos neatskleidžiantys identifikavimo protokolai

Pagrindinė slaptažodžių ir panašių dalinio žinojimo tipo identifikacijos būdų problema yra ta, kad visa ar bent dalis identifikacijai reikalingos informacijos yra perduodama tikrintojui, kuris šią informaciją vėliau gali panaudoti apsimesdamas pareiškėju. Ši problema iškyla net su klausimų-atsakymų protokolais, kurie identifikacijos informacija gali atiduoti mažomis dalelėmis.

Informacijos neatskleidžiantys (angl. *zero-knowledge*) identifikavimo protokolai gali išspręsti šią identifikavimo informacijos nutekėjimo problemą.

1.2.1. Informacijos neatskleidžiančių protokolų veikimas

Informacijos neatskleidžiantys protokolai yra interaktyvios įrodymo sistemos, kuriose tikrintojas ir pareiškėjas pasikeičia ne vienu pranešimu, tipiškai paremtu atsitiktinu skaičiumi, kuris laikomas paslaptyje. Pareiškėjo tikslas yra įrodyti savo tiesą, šiuo atveju parodyti žinias apie atitinkamą paslaptį. Tikrinantysis arba priima, arba atmeta pareiškėjo tvirtinimą. Informacijos neatskleidžiančiuose protokoluose įrodymas yra ne absoliutus, o tikimybinis. Tai reiškia, kad tikslios identifikacijos šansas yra pakankamai artimas 1. Interaktyvios įrodymo sistemos dėl šios priežasties kartais yra vadinamos įrodymu pagal protokolą.

1.2.2. Pagrindinės informacijos neatskleidžiančių protokolų savybės

- *Užbaigtumo savybė* sako, kad jei A ir B yra tuo, kuo sakosi esą, protokolas turi tai patvirtinti.
- *Patikimumo savybė* sako, kad paslaptis s, kartu su viešai žinomais duomenimis, tenkina atitinkamą polinominio laiko predikatą. Bet kas bandantis apsimetinėti pareiškėju A privalo turėti žinias ekvivalenčias A paslapčiai s (šios žinios gali būti išgautos per polinominį laiką). Ši savybė garantuoja, kad protokolas tikrai patvirtina žinias, ekvivalenčias toms, kurios reikalingos prisijungimui.
- *Žinių neatskleidimo savybė* sako, kad joks apsimetelis C dalyvaujantis identifikacijos procese negauna informacijos apie A ar B, kuri nebuvo vieša prieš vykdant protokolą, išskyrus tai, kad A gali įrodyti, kad turi žinias apie paslaptį s. [MOV96, §10.4.1]

1.2.3. Informacijos neatskleidžiantys ir kiti protokolai

Palyginsime informacijos neatskleidžiančius su kitais protokolais:

- *Nėra degradacijos naudojant:* protokolai, turintys informacijos neatskleidimo savybę, netampa lengviau nulaužiami, kai yra dažnai naudojami, nes nesidalinama paslaptimi. Taip pat šie protokolai yra atsparesni parinkto žodyno atakoms.
- *Vengiamas šifravimas:* dauguma informacijos neatskleidžiančių protokolų vengia naudoti aiškius šifravimo algoritmus.
- *Efektyvumas:* nors kai kurios informacijos neatskleidžiančiais protokolais remtos technologijos pasižymi dideliu efektyvumu, protokolai pasižymintys informacijos neatskleidimo savybe dažniausiai turi didesnius komunikacinius ar/ir skaičiavimo kaštus nei kiti algoritmai. Skaičiavimo

efektyvumas praktiškesniuose informacijos neatskleidimo savybe remtuose protokoluose atsiranda iš jų interakyvaus įrodymo, o ne informacijos neatskleidimo savybės aspektų.

- *Neįrodytos prielaidos:* dauguma informacijos neatskleidžiančių protokolų, remiasi tokiomis pačiomis neįrodytomis prielaidomis, kaip ir viešų raktų technologijos. [MOV96, §10.4.1]

1.3. Klaidas taisantys kodai

Nagrinėjant identifikavimo schemas, naudojančias klaidas taisančius kodus, yra būtina šiek tiek susipažinti su greta šios temos naudojamomis temomis, tokiomis kaip klaidas taisantys kodai, tiesiniai kodai, sindromai.

1.3.1. Bendra informacija apie klaidas taisančius kodus

Perduodant informaciją kanalais gali atsirasti įvairūs duomenų iškraipymai. Norint sumažinti informacijos iškraipymo galimybę buvo pradėti naudoti tam tikri metodai padedantys aptikti ir ištaisyti informacijos iškraipymus. Prieš perduodant į kanalą informacija yra koduojama (paruošiama siuntimui), o išėjusi iš kanalo dekoduojama. Kodavimo metu prie norimo siųsti pranešimo yra pridedama papildoma informacija, kuri leidžia aptikti ir ištaisyti tam tikrą skaičių klaidų, padarytų siuntimo metu. Taip gaunamas užkoduotas pranešimas, kuris yra ilgesnis nei pradinis pranešimas. Dekodavimo metu naudojantis papildoma informacija, kuri buvo pridėta kodavimo metu, yra aptinkamas ir, jeigu galima, ištaisomas tam tikras klaidų skaičius. Jeigu kanale padarytų klaidų skaičius nėra didesnis už klaidų skaičių, kurį gali ištaisyti naudojamas klaidas taisantis algoritmas, dekoduotas pranešimas bus yra lygus pradiniam pranešimui.

1.3.1.1. Pagrindinės klaidas taisančių kodų sąvokos

Kalbant apie klaidas taisančius kodus, yra būtina žinoti kodo apibrėžimą. Tarkime, A yra abėcėlė, $|A| = q$. Žymėsime: $A = A_q$, $A_q^n = A_q \times A_q \times \ldots \times A_q$.

Kodas - (n, N) kodu iš abėcėlės A_q žodžių vadinamas bet koks poaibis $C \subset A_q^n$, čia $|C| = N$, N – žodžių skaičius, n – jų ilgis.

Norima perduoti informacija yra suskaidoma į k ilgio vektorius , kurie yra užkoduojami ir tampa n ilgio vektoriais, $n \geq k$.

Kodo koeficientas – santykis $\frac{k}{n}$, tai yra pradinio ir persiunčiamo pranešimų ilgių santykis.

Kodo koeficientas parodo, kuri į kanalą siunčiamų ženklų dalis yra naudinga, o kuri pridėta tik klaidų aptikimui ir ištaisymui. Kuo kodo koeficientas didesnis, tuo geriau, nes gaunama daugiau naudingos informacijos, tačiau kuo kodo koeficientas mažesnis, tuo didesnė tikimybė ištaisyti daugiau klaidų, nes pridedama daugiau informacijos klaidų taisymui.

Dekoduojant yra labai svarbus atstumas tarp žodžių (vektorių), nes gautas žodis yra dekoduojamas tuo kodo elementu, kuris pagal vektorių atstumą yra arčiausias dekoduojamam žodžiui. Paprasčiausias būdas suskaičiuoti atstumą tarp žodžių yra Hamingo atstumas.

Hamingo atstumas – tegu $\mathrm{x} = x_1 \dots x_n$, $\mathrm{y} = y_1 \dots y_n$ yra du aibės A_q^n žodžiai. Hamingo atstumu tarp x ir y vadinsime dydį

$$h(\mathrm{x},\mathrm{y}) = \sum_{\substack{i=1,\dots,n \\ x_i \neq y_i}} 1$$

tai yra skirtingų koordinačių skaičius lyginant du vektorius.

Hamingo atstumo savybės aibėje A_q^n:

1. $h(\mathrm{x},\mathrm{x}) = 0, \mathrm{x} \in A_q^n$.
2. $h(\mathrm{x},\mathrm{y}) = h(\mathrm{y},\mathrm{x}), \mathrm{x},\mathrm{y} \in A_q^n$.
3. $h(\mathrm{x},\mathrm{y}) \leq h(\mathrm{x},\mathrm{z}) + h(\mathrm{y},\mathrm{z}), \mathrm{x},\mathrm{y},\mathrm{z} \in A_q^n$.

Hamingo atstumas yra metrika. [Sta02a, §1.1.1]

Minimalaus atstumo taisyklė - dekodavimo taisyklę $f\colon A_q^n \to C$ vadinsime minimalaus atstumo taisykle, jei su kiekvienu $\mathrm{d} \in A_q^n$

$$\mathrm{h}(\mathrm{d}, f(\mathrm{d})) = \min_{\mathrm{c} \in C} h(\mathrm{d},\mathrm{c})$$

Jei dekoduojamas žodis yra nutolęs vienodu atstumu (pagal Hamingo atstumą) nuo kelių kodo žodžių tai pasirenkamas vienas iš dviejų variantų, arba tiesiog pažymėti klaidą, arba dekoduoti vienu iš galimų variantų.

Dekodavimas gali ištaisyti tam tikrą skaičių klaidų atsiradusių siunčiant informaciją kanalu. Tas skaičius priklauso nuo naudojamo kodo. Galima surasti skaičių klaidų, kuris tikrai bus ištaisytas, bet negalima pasakyti kiek tiksliai klaidų kodas ištaisys. Rasti skaičių klaidų, kurios visada bus ištaisytos, naudojamas kodo minimalus atstumas d. Dekoduojant informacija bus atkurta teisingai, jei kanale įvyko mažiau klaidų nei $\frac{d}{2}$, jei kanale įvyko lygiai $\frac{d}{2}$ klaidų ar daugiau, tai dekodavimas gali būti teisingas arba klaidingas.

Kodo minimalus atstumas – kodo C minimaliu atstumu vadinsime dydį

$$d(C) = \min_{\substack{\mathrm{c},\mathrm{d} \in C \\ \mathrm{c} \neq \mathrm{d}}} h(\mathrm{d},\mathrm{c})$$

Tai yra mažiausias Hamingo atstumas tarp dviejų kodo C žodžių. Jei (n, N) kodo C minimalus atstumas yra d, tai kodą vadinsime (n, N, d) kodu.

t *klaidų taisantis kodas* – jei, naudojant minimalaus atstumo dekodavimo taisyklę, visada dekoduojama teisingai, kai siųstame pranešime yra ne daugiau kaip t klaidų, tai kodą C vadinsime t klaidų taisančiu kodu. [Ske05, §1.5.16]

Tiksliai t *klaidų taisantis kodas* – t klaidų taisantį kodą vadinsime tiksliai t klaidų taisančiu kodu, jei jis nėra $t + 1$ klaidą taisantis kodas. [Ske05, §1.5.16]

Bet koks (n, N, d) kodas taiso tiksliai $\left\lfloor \frac{d-1}{2} \right\rfloor$ klaidų.

Kai kuriais atvejais pakanka tik rasti, kad gautame pranešime yra klaidų.

t klaidų aptinkantis kodas – kodą C vadinsime t klaidų aptinkančiu kodu, jei bet kuriame kodo žodyje įvykus ne daugiau kaip t klaidų, gautas vektorius nepriklauso kodui C. [Ske05, §1.5.16]

Tiksliai t klaidų aptinkantis kodas – t klaidų aptinkantį kodą vadinsime tiksliai t klaidų aptinkančiu kodu, jei jis nėra $t+1$ klaidą aptinkantis kodas. [Ske05, §1.5.16]

Bet koks (n, N, d) kodas aptinka tiksliai $d-1$ klaidą.

1.3.2. Tiesiniai kodai

Tiesiniai kodai – tai kodai, kuriems suteikta tiesinės erdvės struktūra. Tai palengvina jų kodavimą bei dekodavimą. Kalbant apie tiesiniu kodus naudosime žodžių aibę F_q^n (tai yra n-matė tiesinė erdvė virš kūno F_q).

Tiesinis kodas – kodą L, $L \subset F_q^n$, vadinsime tiesiniu, jei L yra tiesinis F_q^n poerdvis. Jei L dimensija lygi k , o minimalus atstumas d , tai kodą L vadinsime $[n, k, d]$, arba tiesiog $[n, k]^q$, kodu. [Sta02a, §2.1.1]

Kiekvienas $[n, k]$ kodas turi q^k elementų. Tačiau, kad jis būtų pilnai apibrėžtas, nebūtina išrašyti visus q^k žodžius. Pakanka tų žodžių, kurie sudaro L kaip tiesinio F_q^n poerdvio bazę.

Generuojanti matrica – tegu L, $L \subset F_q^n$, yra tiesinis $[n, k]$ kodas. Kūno F_q elementų $k \times n$ matricą G, vadinsime generuojančia kodo L matrica, jei n ilgio žodžiai, gauti išrašant matricos G eilučių elementus, sudaro kodo L bazę. [Sta02a, §2.1.2]

Jeigu žinome $[n, k]$ kodo L generuojančią matricą, galime gauti visus L žodžius, imdami skirtingas eilučių kombinacijas: $L = \{xG : x \in F_q^k\}$ (kur G – generuojanti matrica, xG – žodžio, kaip vektoriaus ir matricos sandauga).

Kontrolinė matrica – tegu L yra tiesinis $[n, k]$ kodas. $(n-k) \times n$ matricą H, kuri tenkina sąlygą $L = \{x : xH^T = O_{1,n-k}\}$ (kur $O_{1,n-k}$ – $1 \times (n-k)$ matrica, sudaryta vien tik iš nulių), vadinsime kodo L kontroline matrica.

Kodo minimalus atstumas – tegu H yra tiesinio kodo L kontrolinė matrica. Jeigu egzistuoja d tiesiškai priklausomų H stulpelių, o bet kuri $d-1$ šios matricos stulpelių sistema yra tiesiškai nepriklausoma, tai kodo L minimalus atstumas lygus d.

1.3.2.1. Tiesinių kodų dekodavimas, taikant minimalaus atstumo taisyklę

Tegu $L \subset F_q^n$ yra tiesinis $[n, k]$ kodas. Suskaidysime erdvę F_q^n aibėmis $L_x = x + L$; čia $x \in F_q^n$. Aibės L_x, L_y arba nesikerta, arba sutampa; čia $x, y \in F_q^n$. Aibė $\frac{F_q^n}{L} = \{L_x : x \in F_q^n\}$ yra tiesinė erdvė, gauta faktorizavus F_q^n pagal poerdvį L. [Sta02a, §2.2.3]

Taikydami minimalaus atstumo taisyklę, gautą žodį x dekoduosime žodžiu c pagal formulę $h(c, x) = w(x-c) = \min_{c' \in L} w(x-c')$. Tačiau, žodis $a = x - c$, kaip ir x, yra toje pačioje L_b klasėje. Iš to gauname, kad klasėje L_b reikia rasti mažiausią svorį turintį elementą a ir dekoduoti pagal formulę $x \to f(x) = x - a$.

Dekodavimo procedūra gali būti atlikta taip. Sunumeruojame kodo L žodžius taip, kad žodis $0 = 00 \ldots 0$ būtų pirmas: $L = \{c_0, c_1, \ldots, c_N\}$, $c_0 = 0$, $N = q^k - 1$. Visus F_q^n elementus išrašysime matricoje:

$$\begin{pmatrix} a_0 & c_1 & c_2 & \cdots & c_N \\ a_1 & a_1 + c_1 & a_1 + c_2 & \cdots & a_1 + c_N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_s & a_s + c_1 & a_s + c_2 & \cdots & a_s + c_N \end{pmatrix}$$

Pirmojo stulpelio žodžius a_i parenkame taip, kad jie tenkintų sąlygas:

$$a_0 = 0, w(a_i) = \min(w(a): a \in F_q^n, a \notin \bigcup_{j<i} L_{a_j}\}, j > 1.$$

Šis matricos sudarymo būdas garantuoja, jog kiekvienoje eilutėje išrašyti atitinkamos klasės L_a elementai, o pirmasis iš jų turi mažiausią svorį. Sudarytą matricą vadinsime standartine kodo L lentele, o žodžius a_i – atitinkamų klasių lyderiais.

Susidarius standartinę kodo lentelę, dekodavimo algoritmas tampa toks:

- Randame, kurioje eilutėje gaunamas x.
- x dekoduojame žodžiu $f(x) = x - a$, kur a eilutės lyderis.

1.3.3. Sindromai

Sindromai labai praverčia tiesinių kodų dekodavime, nes jų pagalba lengva nustatyti kuriai klasei priklauso vektorius.

Sindromas – tegu H yra tiesinio kodo C kontrolinė matrica, $y \in F_q^n$. Žodžio y sindromu vadinsime vektorių $s(y) = Hy^T \in F_q^{n-k}$.

Sindromų savybės:

1. Vektorius $y \in C$ tada ir tik tada, kai $s(y) = 0$.
2. Du vektoriai priklauso tai pačiai klasei tada ir tik tada, kai jų sindromai lygūs.
3. Tegu $q = 2$ (tai yra C – dvinaris kodas), o y yra iš kanalo gautas galbūt iškraipytas vektorius. Tada $s(y)$ yra lygus sumai tų kontrolinės matricos H stulpelių, kur įvyko klaidos. [Ske05, §2.5.10]

Iš savybių matome, kad tarp klasių aibės ir sindromų aibės egzistuoja abipusiškai vienareikšmė atitiktis. Taigi norint dekoduoti galime apsibrėžti sumažintą standartinę lentelę:

Klasių lyderiai	**Sindromai**
0	0
a_1	s_1
a_2	s_2
⋮	⋮
a_{r-1}	s_{r-1}

Gavę iš kanalo vektorių y, apskaičiuojame jo sindromą $s(y)$, randame $s_i = s(y)$ sumažintoje standartinėje lentelėje, nusprendžiame, kad atitinkamas klasės lyderis a_i yra klaidų vektorius ir dekoduojame vektorių y kodo žodžiu $y - a_i$.

Sindrominio dekodavimo problema – tai uždavinys rasti vektorių e ilgio n ir svorio p, tokį kad $i = He^t$, kai $H_{(k,n)}$ yra porų tikrinimo matrica dvejetainiam kodui $[n, k]$ ir i turimas sindromas. [MM12, §1.1]

2. Susipažinimas su schemomis

2.1. Stern schema

1994 metais J. Stern pasiūlė naują identifikavimo schemą, paremtą sindrominio dekodavimo problema, turinčią informacijos neatskleidimo savybę. Jo pasiūlyta schema naudoja fiksuotą matricą $H_{[(N-k)\times N]}$ virš dviejų elementų lauko. Ši matrica bendra visiems vartotojams ir pati yra sugeneruojama atsitiktiniu būdu. Ji gali būti laikoma porų tikrinimo matrica, kuri suteikia lengvai taisomą tiesinį dvejetainį kodą.

2.1.1. Identifikavimo schemos veikimas

Bet kuris vartotojas gauna slaptą raktą s, kuris yra n bitų žodis su iš anksto nustatytu svoriu (p). Skaičius p yra sistemos dalis. Viešasis identifikavimo raktas skaičiuojamas formule $i = H(s)$ (čia $H(s)$ – tai matricos H sandauga su iš s pasigamintu vektoriumi-stulpeliu). [Ste94, §1]

Identifikacijos schema remiasi įsipareigojimo idėja. Jei u yra bitų seka, tai įsipareigojimas u, yra u vaizdas perleidus seką per kriptografinę maišos funkciją. Įsipareigojimas yra vienpusė funkcija, todėl norint ją atskleisti, reikia atskleisti originalią seką iš kurios ji buvo pagaminta. Kai tai padaryta, įsipareigojimo teisingumą gali patikrinti bet kas.

Ši identifikavimo schema naudoja elementarų interaktyvų informacijos neatskleidžiantį protokolą. Informacijos neatskleidžiantis protokolas leidžia bet kuriam pareiškėjui identifikuoti save tikrintojui. Protokolo vykdymą apima r iteracijų, kurios vykdomos taip:

Claimer C		**Verifier V**
$SK(s), PK(G,i)$		$PK(G,i)$
$y \leftarrow F_2^n, o \leftarrow S_n$		
$c_1 \leftarrow h(o, H(y))$		
$c_2 \leftarrow h(y.o)$		
$c_3 \leftarrow h((y \oplus s).o)$		
	$\xrightarrow{c_1\ c_2\ c_3}$	$b \leftarrow \{0,1,2\}$
	$\xleftarrow{b}$	
Case $b = 0$	$\xrightarrow{y\ o}$	Check $c_1\ c_2$
Case $b = 1$	$\xrightarrow{y \oplus s\ o}$	Check $c_1\ c_3$
Case $b = 2$	$\xrightarrow{y.o\ s.o}$	Check $c_2\ c_3$

1. Pareiškėjas išsirenka atsitiktinį n bitų žodį y, kartu su atsitiktine permutacija o sudaryta iš sveikųjų skaičių $\{1 \ldots n\}$.
2. Tada pareiškėjas išsiunčia įsipareigojimus c_1, c_2, c_3 atitinkamai $h(o, H(y))$, $h(y.o)$ ir $h((y \oplus s).o)$, tikrintojui. Svarbu žinoti, kad kablelis maišos funkcijoje $h(,)$ reiškia maišos funkcijos naudojimą visų informacijos bitų pateiktų argumentuose sąjungai. Permutacija o šiuo atveju yra verčiama y ir $y \oplus s$ elementus koduojančių bitų vektoriumi. Taip pat svarbu žinoti, kad $y.o$ nurodo y atvaizdą po o permutacija.
3. Tikrintojas atsako atsiusdamas atsitiktinį elementą b iš aibės $\{0,1,2\}$.
4. Jei b yra 0, pareiškėjas atskleidžia y ir o.
   Jei b yra 1, atskleidžiama $y \oplus s$ ir o.
   Jei b yra 2 pareiškėjas atskleidžia $y.o$ ir $s.o$.
5. Jei b yra 0, tikrintojas peržiūri ar teisingai suskaičiuoti įsipareigojimai c_1 ir c_2.
   Jei b yra 1, tikrintojas peržiūri įsipareigojimus c_1 ir c_3 ($H(y)$ randamas iš $H(y \oplus s)$ formule $H(y) = H(y \oplus s) \oplus i$).
   Jei b yra 2, tikrintojas patikrina įsipareigojimus c_2 ir c_3. [Ste94, §1]

2.1.2. Programos veikimas

Programa veiksmams su vektoriais ir pačių vektorių saugojimui naudoja klasę BitSeq, matricoms ir veiksmams su jomis naudojama klasė Matrix. Programa tik įjungus sugeneruoja pareiškėjo paslaptį s, bei matricą H. Pagal sugeneruotus duomenis yra suskaičiuojama likusi viešojo rakto dalis – i. Susigeneravus

viešąjį ir privatų raktus pagal juos sukuriamos pareiškėjo (Claimer) ir tikrintojo(Verifier) šalys, reprezentuojamos atitinkamų klasių, sukuriamas ir pradedamas vykdyti pareiškėjo tikrinimo procesas.

Tikrinimo proceso metu vykdomos atskiros Stern schemos iteracijos: atsitiktiniu būdu sugeneruojami iteracijos metu naudojamas vektorius y ir permutacija o, atvaizduojama Permutation klase, paruošiami bei tikrintojui perduodami įsipareigojimai.

Gavęs įsipareigojimus tikrintojas siunčia užklausą b, išreikštą atsitiktiniu skaičiumi iš aibės $\{0, 1, 2\}$, į kurią pareiškėjas atsako atskleisdamas iteracijai sugeneruotus atsitiktinius duomenis bei jų kombinacijas su saugoma paslaptimi s.

Turėdamas atsitiktinius duomenis tikrintojas naujai suskaičiuoja atitinkamus įsipareigojimus ir grąžina teigiamą arba neigiamą atsakymą priklausomai nuo to, ar suskaičiuoti duomenys atitinka ankstesnius įsipareigojimus.

Jeigu tikrintojas kiekvienos iteracijos metu grąžina teigiamą reikšmę, pareiškėjas priimamas.

2.1.2.1. Programos išvesties vienai iteracijai pavyzdys

Programa vienos iteracijos metu atspausdina atsitiktinius duomenis sugeneruotus iteracijai, bei komunikaciją tarp pareiškėjo ir tikrintojo.

Pradiniai programos kintamieji: žodžio *s* ilgis *n = 256*, svoris *p = 43*

S = 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0
0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
I = 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 1 1 0 1 0 1 1
0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 0 1 1 1 1 0 1 1
1 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 0 0 1 0 1 1 1 1 0 0 1 1 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0
0 0 0 0 1 1 1 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0
===CLAIMER===
Y: 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0
1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0
O: 237 96 102 110 128 80 106 76 87 41 125 126 68 49 133 140 10 82 54 174 158 175 180 93 89 183 189 23 48 15 66 145 166 56 192 149 194 200 148 36 207 209 181 211 123 46 203 219 222 156 157 70 111 121 84 155 169 224 227 212 235 238 171 228 0 83 79 78 65 118 122 26 116 124 132 179 191 198 202 221 239 5 240 90 241 115 245 24 92 113 37 57 42 91 114 53 137 6 25 138 139 20 27 3 152 159 4 45 182 16 190 117 17 61 43 232 13 130 127 150 14 63 108 71 101 131 88 147 216 44 99 187 246 247 112 55 176 141 197 104 178 210 19 64 226 62 231 33 154 243 146 12 103 67 107 120 136 160 168 38 50 31 52 173 193 163 208 213 186 51 105 29 188 95 47 135 201 7 215 40 58 69 142 161 217 185 218 220 234 162 72 223 242 248 249 143 8 205 11 98 1 165 39 97 214 236 18 74 22 32 134 129 151 153 2 75 229 59 177 250 251 167 230 34 35 60 206 196 244 21 199 252 253 254 100 225 233 255 73 86 9 109 184 144 81 164 94 170 85 30 119 172 195 204 28 77
---COMMITMENTS--
C1: -674626056
C2: 2037241279

C3: 1309244283

---B----------

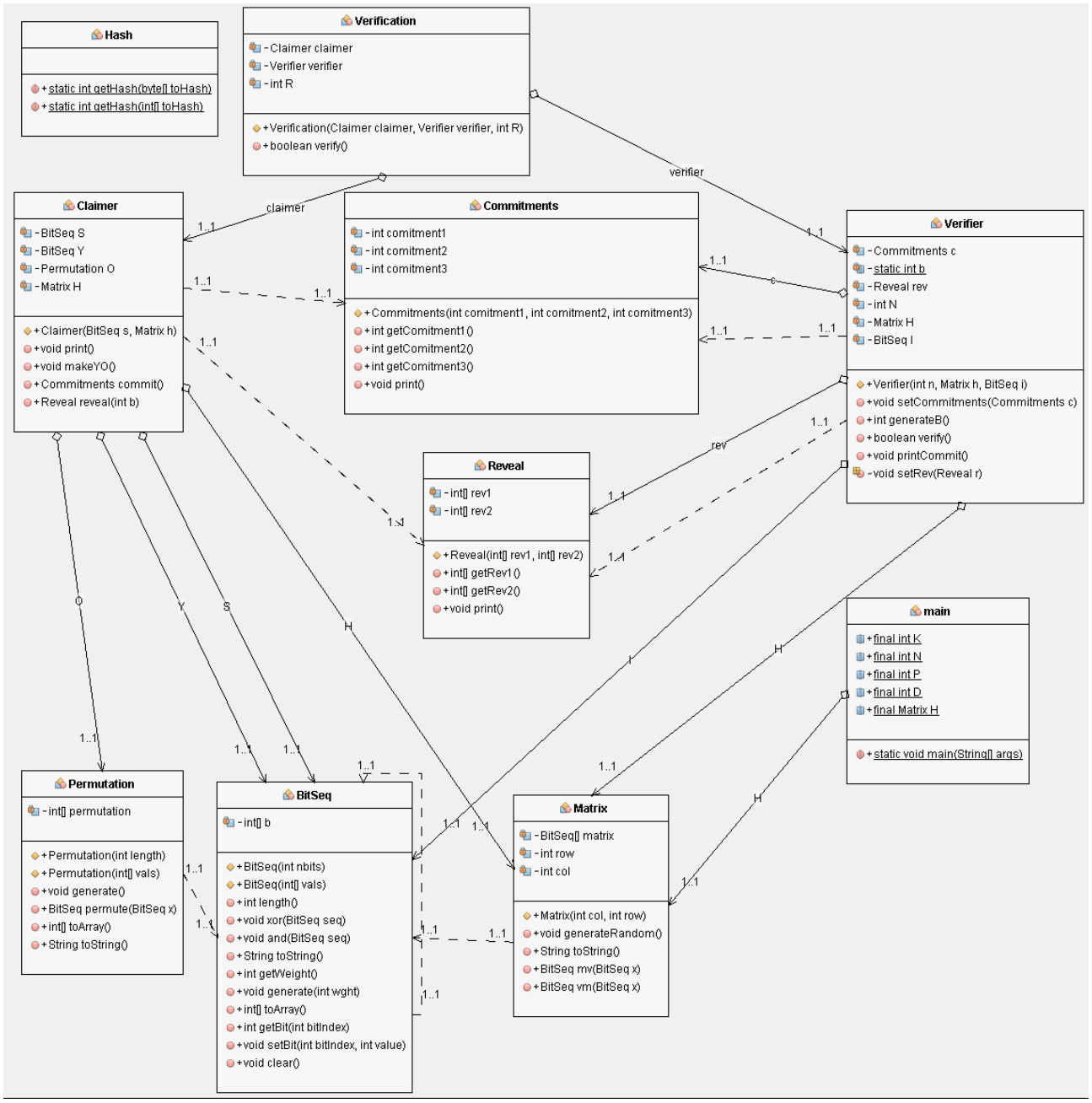
B: 1

---REVEALS----------

R1: 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0

R2: 237 96 102 110 128 80 106 76 87 41 125 126 68 49 133 140 10 82 54 174 158 175 180 93 89 183 189 23 48 15 66 145 166 56 192 149 194 200 148 36 207 209 181 211 123 46 203 219 222 156 157 70 111 121 84 155 169 224 227 212 235 238 171 228 0 83 79 78 65 118 122 26 116 124 132 179 191 198 202 221 239 5 240 90 241 115 245 24 92 113 37 57 42 91 114 53 137 6 25 138 139 20 27 3 152 159 4 45 182 16 190 117 17 61 43 232 13 130 127 150 14 63 108 71 101 131 88 147 216 44 99 187 246 247 112 55 176 141 197 104 178 210 19 64 226 62 231 33 154 243 146 12 103 67 107 120 136 160 168 38 50 31 52 173 193 163 208 213 186 51 105 29 188 95 47 135 201 7 215 40 58 69 142 161 217 185 218 220 234 162 72 223 242 248 249 143 8 205 11 98 1 165 39 97 214 236 18 74 22 32 134 129 151 153 2 75 229 59 177 250 251 167 230 34 35 60 206 196 244 21 199 252 253 254 100 225 233 255 73 86 9 109 184 144 81 164 94 170 85 30 119 172 195 204 28 77

2.1.2.2. Programos UML diagrama



2.2. Veron schema

1997 metais P. Veron pasiūlė naujų modifikacijų klasikinei J. Stern schemai. Jo pasiūlyta schema taip pat remiasi sindromų dekodavimo problema, tačiau pasiūlyti pokyčiai turėjo sumažinti komputacinę apkrovą bei saugomų duomenų kiekį pareiškėjo pusėje. Schema naudoja atsitiktinai sugeneruotą matricą $G_{[k \times N]}$ virš dviejų elementų lauko. Ši matrica yra bendra visiems naudotojams.

2.2.1. Identifikavimo schemos veikimas

Bet kuris vartotojas gauna slaptus raktus e ir m, kurie yra n ir k bitų ilgio žodžiai atitinkamai. Viešieji identifikavimo raktai yra e svoris (žymimas p) ir $x = mG + e$.

Identifikacijos schema taip pat remiasi įsipareigojimo idėja.

Protokolo vykdymą apima r iteracijų, kurios vykdomos taip:

Claimer C		**Verifier V**
$SK(e,\ m), PK(H, x, p)$		$PK(H, x,\ p)$
$u \leftarrow F_2^k, o \leftarrow S_n$		
$c_1 \leftarrow h(o)$		
$c_2 \leftarrow h((u \oplus m)G.o)$		
$c_3 \leftarrow h((uG \oplus x).o)$		
	$\xrightarrow{c_1\ c_2\ c_3}$	$b \leftarrow \{0,1,2\}$
	$\xleftarrow{b}$	
Case $b = 0$	$\xrightarrow{u \oplus m\ o}$	Check $c_1\ c_2$
Case $b = 1$	$\xrightarrow{(u \oplus m)G.o\ \ e.o}$	Check $c_2\ c_3$
Case $b = 2$	$\xrightarrow{u\ o}$	Check $c_1\ c_3$

1. Pareiškėjas išsirenka atsitiktinį k bitų žodį u, kartu su atsitiktine permutacija o sudaryta iš sveikųjų skaičių $\{1 \ldots n\}$. Tada išsiunčia įsipareigojimus c_1, c_2, c_3 atitinkamai $h(o)$, $h((u + m)G.o)$ ir $h((uG + x).o)$, tikrintojui.
2. Tikrintojas atsako atsiųsdamas atsitiktinį elementą b iš aibės $\{0,1,2\}$.
3. Jei b yra 0, pareiškėjas atskleidžia $u + m$ ir o.
   Jei b yra 1, atskleidžiama $(u + m)G.o$ ir $e.o$.
   Jei b yra 2 pareiškėjas atskleidžia o ir u.
4. Jei b yra 0, tikrintojas peržiūri ar teisingai suskaičiuoti įsipareigojimai c_1 ir c_2.
   Jei b yra 1, tikrintojas peržiūri įsipareigojimus c_2 ir c_3 $(uG + x).o$ randamas iš $e.o$ ir $(u + m)G.o$ pagal formule $(uG + x).o = e.o + (u + m)G.o$.
   Jei b yra 2, tikrintojas patikrina įsipareigojimus c_1 ir c_3. [Ver97, § 2]

2.2.2. Programos veikimas

Programa kaip ir prieš tai aptarta Stern schemos interpretacija veiksmams su vektoriais ir pačių vektorių saugojimui naudoja klasę BitSeq, matricoms ir veiksmams su jomis naudoja klasė Matrix. Maišos funkcijai naudojama klasė Hash. Programa tik įjungus sugeneruoja pareiškėjo paslaptis e ir m bei viešuosius

duomenis: matricą G ir vektorių x .pagal šiuos duomenis yra sukuriami pareiškėjo (Claimer) ir tikrintojo (Verifier) klasių atstovai, inicijuojamas pareiškėjo tikrinimo procesas.

Tikrinimo proceso metu vykdomos atskiros Veron schemos iteracijos: atsitiktiniu būdu sugeneruojami iteracijos metu naudojamas vektorius u ir permutacija o, paruošiami bei tikrintojui perduodami įsipareigojimai.

Gavęs įsipareigojimus tikrintojas siunčia užklausą b, Atsitiktinį skaičių iš aibės $\{0, 1, 2\}$, į kurią pareiškėjas atsako atskleisdamas iteracijai sugeneruotus atsitiktinius duomenis bei jų kombinacijas su saugomomis paslaptimis e ir m.

Turėdamas atsitiktinius duomenis tikrintojas naujai suskaičiuoja atitinkamus įsipareigojimus ir grąžina teigiamą arba neigiamą atsakymą priklausomai nuo to, ar suskaičiuoti duomenys atitinka ankstesnius įsipareigojimus.

Jeigu tikrintojas kiekvienos iteracijos metu grąžina teigiamą reikšmę, pareiškėjas priimamas.

2.2.2.1. Programos išvesties vienai iteracijai pavyzdys

Programa vienos iteracijos metu atspausdina atsitiktinius duomenis sugeneruotus iteracijai, bei komunikaciją tarp pareiškėjo ir tikrintojo.

Pradiniai programos kintamieji: žodžių *e* ir *m* ilgiai *n = 256*, svoris *p = 42*
E = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 M = 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 0 0 ===CLAIMER=== X: 1 0 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 1 0 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1 U: 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 O: 67 80 116 65 130 137 13 55 129 138 25 40 147 158 5 167 168 176 109 111 93 57 157 106 100 174 177 99 54 83 21 175 63 178 120 182 114 186 118 33 18 31 35 36 58 102 43 19 71 194 98 104 76 200 191 115 201 204 212 217 218 53 132 70 117 185 221 222 96 227 228 92 10 229 183 197 230 164 232 105 196 235 160 148 238 241 37 242 1 244 3 84 202 50 113 68 192 162 246 77 78 163 46 189 66 190 34 206 245 56 26 165 149 8 247 248 20 142 249 48 107 12 62 128 4 170 9 131 32 150 187 82 140 22 209 139 213 39 27 199 89 136 215 251 141 75 252 125 193 253 166 254 30 88 7 172 101 79 226 255 87 250 44 69 81 86 144 233 15 59 74 95 103 23 123 127 153 207 121 42 179 208 210 220 236 72 237 126 51 171 0 11 29 41 45 146 161 152 28 156 112 180 211 85 119 169 184

52 133 214 216 108 173 219 223 225 205 240 49 234 198 14 243 203 231 134 64 143 154 151 181 47 2 135 239 17 195 94 224 188 38 60 90 6 16 91 97 110 124 145 155 159 122 73 24 61

---COMMITMENTS----------

C1: -2077061313

C2: -1545900671

C3: 1316315233

---B----------

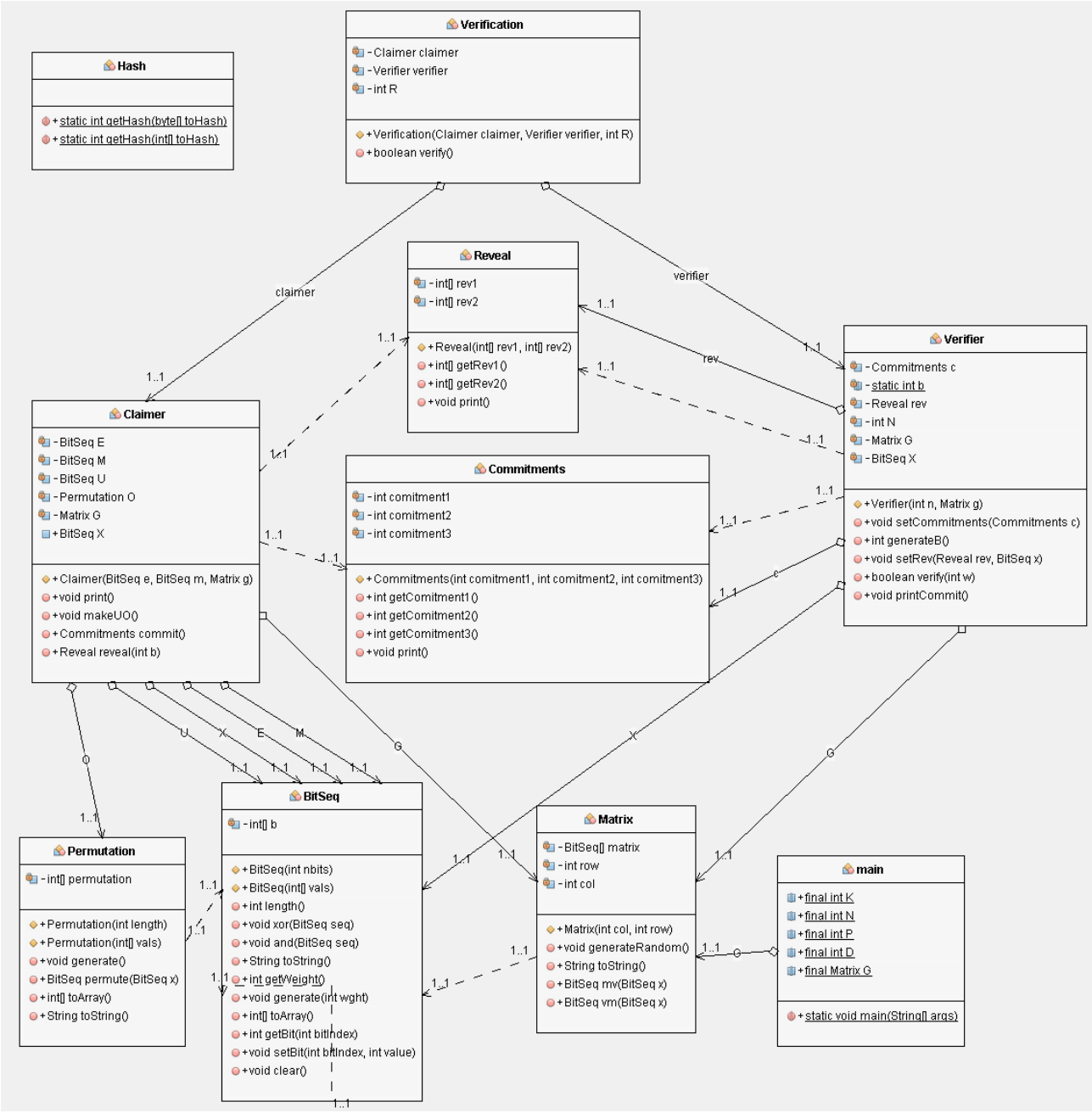
B: 2

---REVEALS----------

R1: 67 80 116 65 130 137 13 55 129 138 25 40 147 158 5 167 168 176 109 111 93 57 157 106 100 174 177 99 54 83 21 175 63 178 120 182 114 186 118 33 18 31 35 36 58 102 43 19 71 194 98 104 76 200 191 115 201 204 212 217 218 53 132 70 117 185 221 222 96 227 228 92 10 229 183 197 230 164 232 105 196 235 160 148 238 241 37 242 1 244 3 84 202 50 113 68 192 162 246 77 78 163 46 189 66 190 34 206 245 56 26 165 149 8 247 248 20 142 249 48 107 12 62 128 4 170 9 131 32 150 187 82 140 22 209 139 213 39 27 199 89 136 215 251 141 75 252 125 193 253 166 254 30 88 7 172 101 79 226 255 87 250 44 69 81 86 144 233 15 59 74 95 103 23 123 127 153 207 121 42 179 208 210 220 236 72 237 126 51 171 0 11 29 41 45 146 161 152 28 156 112 180 211 85 119 169 184 52 133 214 216 108 173 219 223 225 205 240 49 234 198 14 243 203 231 134 64 143 154 151 181 47 2 135 239 17 195 94 224 188 38 60 90 6 16 91 97 110 124 145 155 159 122 73 24 61

R2: 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0

2.2.2.2. Programos UML diagrama



2.3. Harari schema

Kita žinių neatskleidžianti identifikavimo schema, kurią bandysime buvo originaliai pasiūlyta S. Harari. Harari schema taip pat remta sindrominio dekodavimo problema tiesiniams kodams ir atitinka žinių neatskleidimo savybes.

Harari schema naudojasi patikimo šaltinio sugeneruota, atsitiktine $H_{[(n-k)\times n]}$ matrica, virš 2 elementų lauko.

2.3.1. Identifikavimo schemos veikimas

Kiekvienas naudotojas iš patikimo šaltinio gauna savo slaptą raktą s. s tai ilgio N vektorius, turintis fiksuotą svorį μ. μ – nedidelis, nelyginis skaičius iš intervalo $[50, 100]$. Pastebėkime, kad kadangi s – raktas, $Hs^T = 0$. Porų matrica H ir svoris μ yra vieši duomenys.

Identifikavimo schema remiasi atsitiktinių vektorių manipuliacijomis bei vektorių svoriais.

Ši identifikavimo schema naudoja interaktyvų informacijos neatskleidžiantį protokolą. Informacijos neatskleidžiantis protokolas leidžia bet kuriam pareiškėjui identifikuoti save tikrintojui. Protokolo vykdymą apima r iteracijų, kurios vykdomos taip:

Claimer C		**Verifier V**
$SK(s), PK(H, \mu)$		$PK(H, \mu)$
$l \leftarrow [100, 200], R_i \leftarrow F_2^k, i = 1, \dots, l$		
$WS_i \leftarrow \{[w_i, t_i] \mid w_i = w(R_i), t_i = Hr_i^T, i = 1, \dots, l\}$		
	$\xrightarrow{WS}$	$e \leftarrow F_2^l, \frac{l}{3} \leq w(e) \leq \frac{2l}{3}$
	$\xleftarrow{e}$	
	$\xrightarrow{\pi}$	
$\pi \leftarrow S_l$		
$t = \pi(e)$		$t = \pi(e)$
		$b_i \in \{t, \bar{t}\}, i = 1, \dots, l$
	$\xleftarrow{b}$	
	$\xrightarrow{y}$	
$r \leftarrow \sum_{i=1}^{l} b_i r_i, y \leftarrow r + s$		Check:
		$w(y) \neq \sum_{i=1}^{l} b_i w_i$
		$Hy^T = \sum_{i=1}^{l} b_i t_i$.
		$-\mu + \sum_{i=1}^{l} b_i w_i \leq w(y) \leq \mu + \sum_{i=1}^{l} b_i w_i$.

1. Pareiškėjas pasirenka skaičių $l \in [100, 200]$. Tada susigeneruoja l atsitiktinių vektorių $r_i \in F_{2^n}$ kur $i \in \{1, \dots, l\}$, kur vektorių nenulinių koordinačių aibės nesikerta. Sugeneruotų vektorių svorius pasižymime w_i.
2. Pareiškėjas apskaičiuoja sindromus $t_i = Hr_i^T$, kur r_i^T – transponuotas vektorius r_i.
3. Tikrintojas gauna rinkinį $\{[w_i, t_i] \mid i = 1, \dots, l\}$ iš pareiškėjo, tada pasirenka nelyginio svorio vektorių e , kurio ilgis būtų lygus l, ir kuris tenkina savybę: $\frac{l}{3} \leq w(e) \leq \frac{2l}{3}$. Tikrintojas siunčia e pareiškėjui.
4. Pareiškėjas sugeneruoja l narių permutaciją π, ir persiunčia ją tikrintojui.
5. Pareiškėjas ir tikrintojas suskaičiuoja $t = \pi(e)$.
6. Tikrintojas atsitiktinai pasirenka $b \in \{t, \bar{t}\}$, kur $\bar{t}$ yra t dvejetainis papildinys. Kiekvienas b bitas bus apibrėžiamas kaip b_i , kur $i \in \{1, \dots, l\}$. Tikrintojas nusiunčia b pareiškėjui.

7. Pareiškėjas susiskaičiuoja $r = \sum_{i=1}^{l} b_i r_i$, tada tikrintojui siunčia $y = r + s$. Pastaba: veiksmas $b_i r_i$ yra naudojamas pasižymėti, kurie vektoriai iš r_i bus naudojami, priklausomai nuo to ar b_i lygus 1.
8. Tikrintojas patikrina 3 sąlygas galimai pareiškėjo autentifikacijai:
   a. Pirmoji svorio sąlyga: $w(y) \neq \sum_{i=1}^{l} b_i w_i$.
   b. Sindromų sąlyga: $Hy^T = \sum_{i=1}^{l} b_i t_i$.
   c. Antroji svorio sąlyga: $-\mu + \sum_{i=1}^{l} b_i w_i \leq w(y) \leq \mu + \sum_{i=1}^{l} b_i w_i$.

Jei kiekvienoje iš r iteracijų yra patenkinamos visos sąlygos, teigiama kad pareiškėjas pakankamai įrodė savo žinias apie s . [MM12, §1.2]

2.3.2. Programos veikimas

Programa tik įjungus sugeneruoja pareiškėjo paslaptį s išreikštą sveikųjų skaičių masyvu bei matricą H, išreikštą dvimačiu masyvu. Turint šiuos duomenis pagal juos yra sukuriami pareiškėjo (Claimer) ir tikrintojo (Verifier) klasių atstovai, inicijuojamas pareiškėjo tikrinimo procesas.

Tikrinimo procesas vyksta iteracijomis, kurių kiekviena susideda iš šių žingsnių:

Pareiškėjas pasirenka atsitiktinį skaičių l ir tada su tikrinimo matricos pagalba, kad nenulinės koordinatės nesikirstų, sugeneruoja tokį kiekį atsitiktinių vektorių. Pareiškėjas tada apskaičiuoja sugeneruotų vektorių svorius bei sindromus ir šių porų masyvą siunčia tikrintojui.

Tikrintojas sugeneruoja l ilgio vektorių e parinkdamas atsitiktinį nelyginį svorį, ir perduoda jį pareiškėjui.

Pareiškėjas sugeneruoja permutaciją l nariams ir siunčia ją tikrintojui.

Tikrintojas susigeneruoja l ilgio masyvą b ir jį užpildo reikšmėmis iš aibės $\{0,1\}$ gautomis permutuojant anksčiau sugeneruotą vektorių e ir pasirenkant reikšmę priešingą anksčiau turėtai.

Pareiškėjas pagal gautas b reikšmes susigeneruoja masyvą r, kuris yra gaunamas tarpusavyje sudedant anksčiau susigeneruotų vektorių reikšmes jei atitinkamo b elemento reikšmė 1. Tikrintojui tada siunčiama paslapties s ir masyvo r suma.

Galiausia, tikrintojas vykdo surinktų duomenų tikrinimo procedūrą: patikrina ar y svoris nelygus b svoriui, ar H ir y sandauga nėra lygi vektoriui gaunamam iš sindromų pagal turimą masyvą b, ir galiausia ar vektoriaus y svoris yra tarp b svorio pridėjus μ ir b svorio atėmus μ.

Jei visi tikrinimai teigiami, tikrintojas grąžina teigiamą reikšmę.

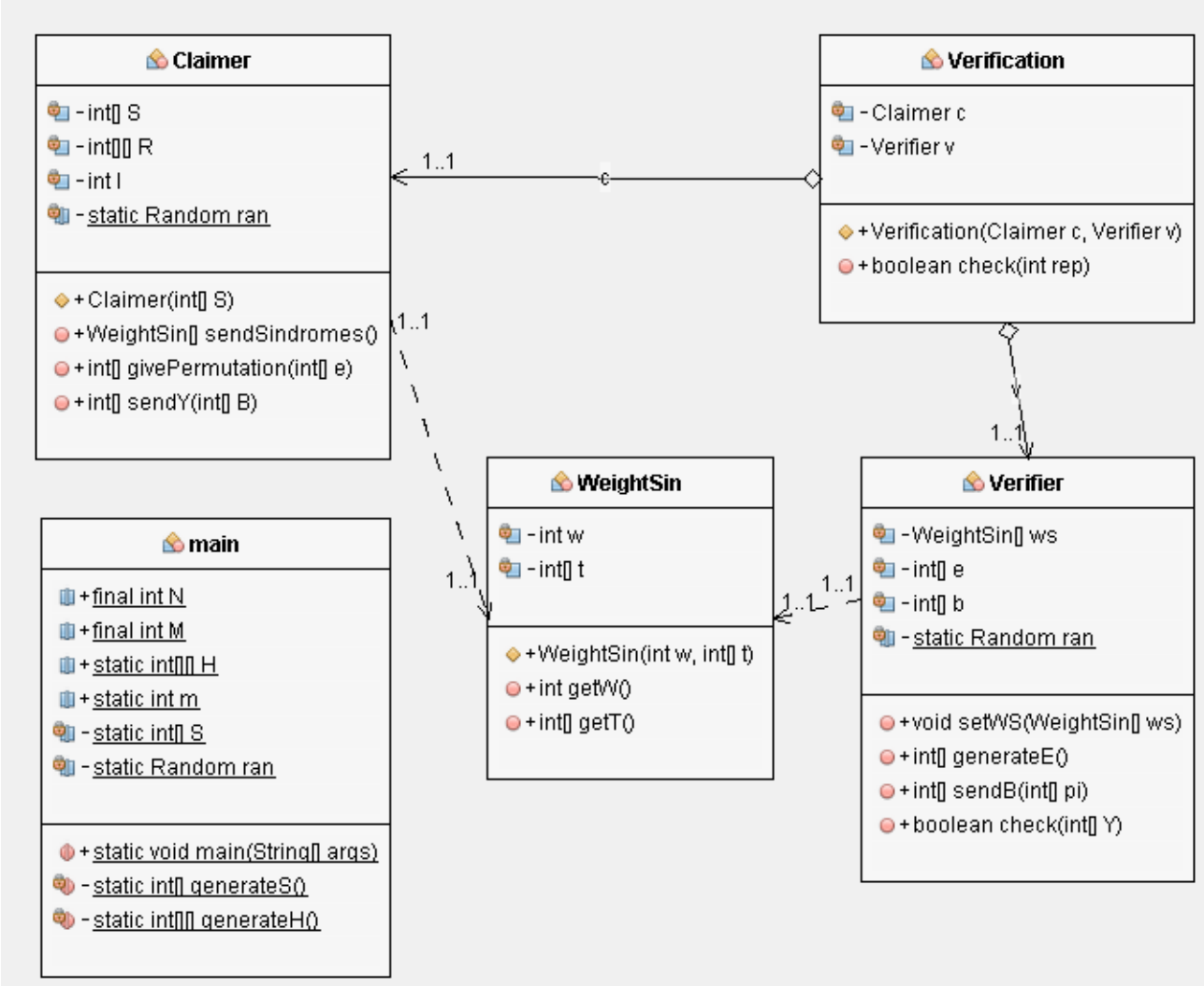
Jeigu tikrintojas kiekvienos iteracijos metu grąžina teigiamą reikšmę, pareiškėjas priimamas.

2.3.2.1. Programos išvesties vienai iteracijai pavyzdys

Harari schemos programa iteracijos metu išveda mažiau duomenų nei Stern ir Veron. Tai buvo pasirinkta, nes Harari schema naudoja masyvias matricas, kurių išvedimas labai apkrautų sistemą.

Pradiniai programos kintamieji: matricos *H* matmenys *n = 256*, *m = 256*
S = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 ==VERIFICATION========== l = 106 --Permutation---------- Pi = 18 31 34 45 6 47 48 59 12 21 61 63 11 51 65 64 30 36 7 42 58 10 32 28 35 43 50 60 14 20 39 67 17 25 56 52 68 29 46 69 70 73 71 62 16 74 76 41 81 82 83 85 86 87 90 91 8 92 27 44 93 94 4 26 38 49 57 89 95 96 19 66 97 2 99 3 53 79 88 101 102 103 54 13 15 78 84 5 33 104 105 77 1 40 55 106 72 75 37 80 9 24 100 98 22 23 Checks passed

2.3.2.2. Programos UML diagrama



3. Tyrimas

Identifikavimo schemų tyrimas buvo padalintas į dvi dalis – pirmiausia buvo ištirtos kaip pasirinktų schemų atskiros iteracijos, tada buvo įgyvendinti schemų vykdymai specifiniam saugumo lygiui ir palyginti gauti rezultatai.

Schemų vertinimui naudojami pasirinkti kriterijai: bendras vykdymo laikas, bei jo pasiskirstymas tarp pareiškėjo ir tikrintojo, operacijų kiekis pareiškėjo ir tikrintojo pusėse, siunčiamų duomenų kiekis.

Vykdymo laikas skaičiuojamas fiksuojant atskirų veiksmų atleikamų pareiškėjo ir tikrintojo trukmes ir jas sumuojant atitinkamai. Operacijų kiekio skaičiavimui sumuojamas kiekis aritmetinių bei loginių operacijų atliekamų pareiškėjo bei tikrintojo. Tyrimo metu visoms aritmetinėms bei loginėms operacijoms priskirtas vienodas svoris.

3.1. Pavienių iteracijų tyrimas

Pavienių schemų tyrimui buvo pasirinkti pradinių duomenų ilgiai $N = 64;\ 256;\ 1024$ ir svoriai $p\ (\mu) = 11;\ 43;\ 171$ (svoriai parinkti pagal Stern schemos rekomendacijas ir sulyginti visoms schemoms). Rezultatams gauti buvo vykdoma 1 000 iteracijų. Tyrimo metu informacijos apie iteraciją išvedimas buvo laikinai atjungtas, nes skirtingas išvedimo laikas bei išvesties kiekis darytų įtaka tyrimo rezultatams.

3.1.1. Stern schema

	$n = 64;\ p = 11$	$n = 256;\ p = 43$	$n = 1024;\ p = 171$
Minimalus vykdymo laikas	0.019243ms	0.174056ms	3.637622ms
Maksimalus vykdymo laikas	2.649955ms	4.348812ms	24.905703ms
Vidutinis vykdymo laikas	0.15539ms	0.366074ms	10.005489ms
Vidutinis laikas pareiškėjo pusėje	0.117327ms	0.243298ms	6.002783ms
Vidutinis laikas tikrintojo pusėje	0.038063ms	0.122776ms	4.002706ms
Vidutinis pareiškėjo operacijų kiekis	299 464	4 736 816	75 570 054
Vidutinis tikrintojo operacijų kiekis	198 441	3 089 728	51 279 255
Vidutinis siunčiamų duomenų kiekis [siuntimų kiekis]	258B [3]	2 064B [3]	8 208B [3]

3.1.2. Veron schema

	$n = 64;\ p = 11$	$n = 256;\ p = 43$	$n = 1024;\ p = 171$
Minimalus vykdymo laikas	0.014539ms	0.117605ms	1.596431ms
Maksimalus vykdymo laikas	4.158507 ms	3.621800ms	9.300616ms
Vidutinis vykdymo laikas	0.060981ms	0.200161ms	2.322944ms

Vidutinis laikas pareiškėjo pusėje	0.047970ms	0.162864ms	1.868932ms
Vidutinis laikas tikrintojo pusėje	0.013010ms	0.037296ms	0.454011ms
Vidutinis pareiškėjo operacijų kiekis	153 768	2 385 380	37 848 921
Vidutinis tikrintojo operacijų kiekis	99 713	1 577 897	25 188 894
Vidutinis siunčiamų duomenų kiekis [siuntimų kiekis]	258B [3]	2 064B [3]	8 208B [3]

3.1.3. Harari schema

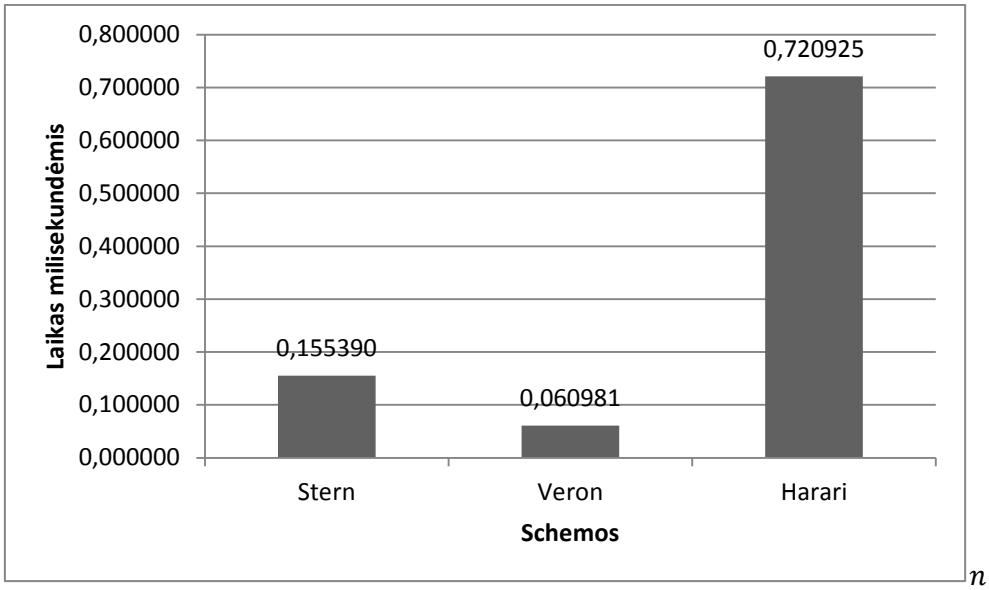
	$n = 64;\ \mu = 11$	$n = 256;\ \mu = 43$	$n = 1024;\ \mu = 171$
Minimalus vykdymo laikas	0.456733ms	24.265505ms	1 013.052848ms
Maksimalus vykdymo laikas	7.158069ms	84.059205ms	2 019.140929ms
Vidutinis vykdymo laikas	0.720925ms	45.300769ms	1 544.139655ms
Vidutinis laikas pareiškėjo pusėje	0.691546ms	45.062213ms	1 538.245662ms
Vidutinis laikas tikrintojo pusėje	0.029379ms	0.238556ms	5.893993ms
Vidutinis pareiškėjo operacijų kiekis	1 276 298	59 824 421	974 383 958
Vidutinis tikrintojo operacijų kiekis	36 178	516 120	6 773 124
Vidutinis siunčiamų duomenų kiekis [siuntimų kiekis]	13 856B [5]	157 024B [5]	620 896B [5]

3.1.4. Palyginimas

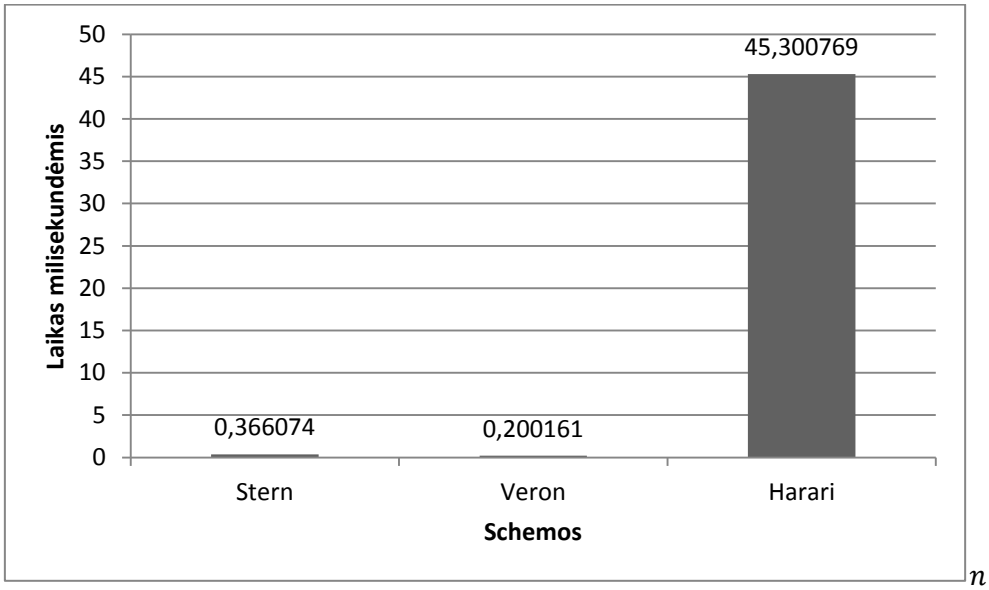
Atskiros schemų iteracijos buvo palygintos pagal vidutinius vykdymo laikus, darbo pasiskirstymą tarp pareiškėjo ir tikrintojo (žiūrimas laiko ir operacijų kiekio pasiskirstymas), bei pagal siunčiamus duomenis.

3.1.4.1. Laiko palyginimas

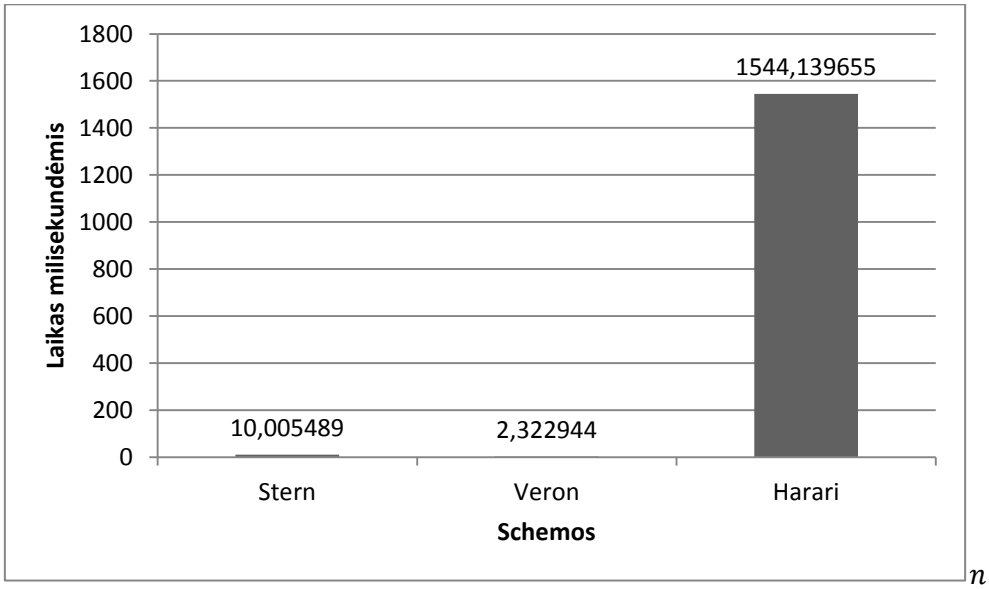
Lygindami iteracijų atlikimo laiką matome, kad Veron schemos iteracijos atliekamos greičiausia. Lėčiausia vykdoma Harari schemos iteracija kurios vykdymo laikas gerokai peržengia Veron bei Stern. Šios tendencijos auga didėjant žodžių ilgiui.



$n = 64;\ p(\mu) = 11$

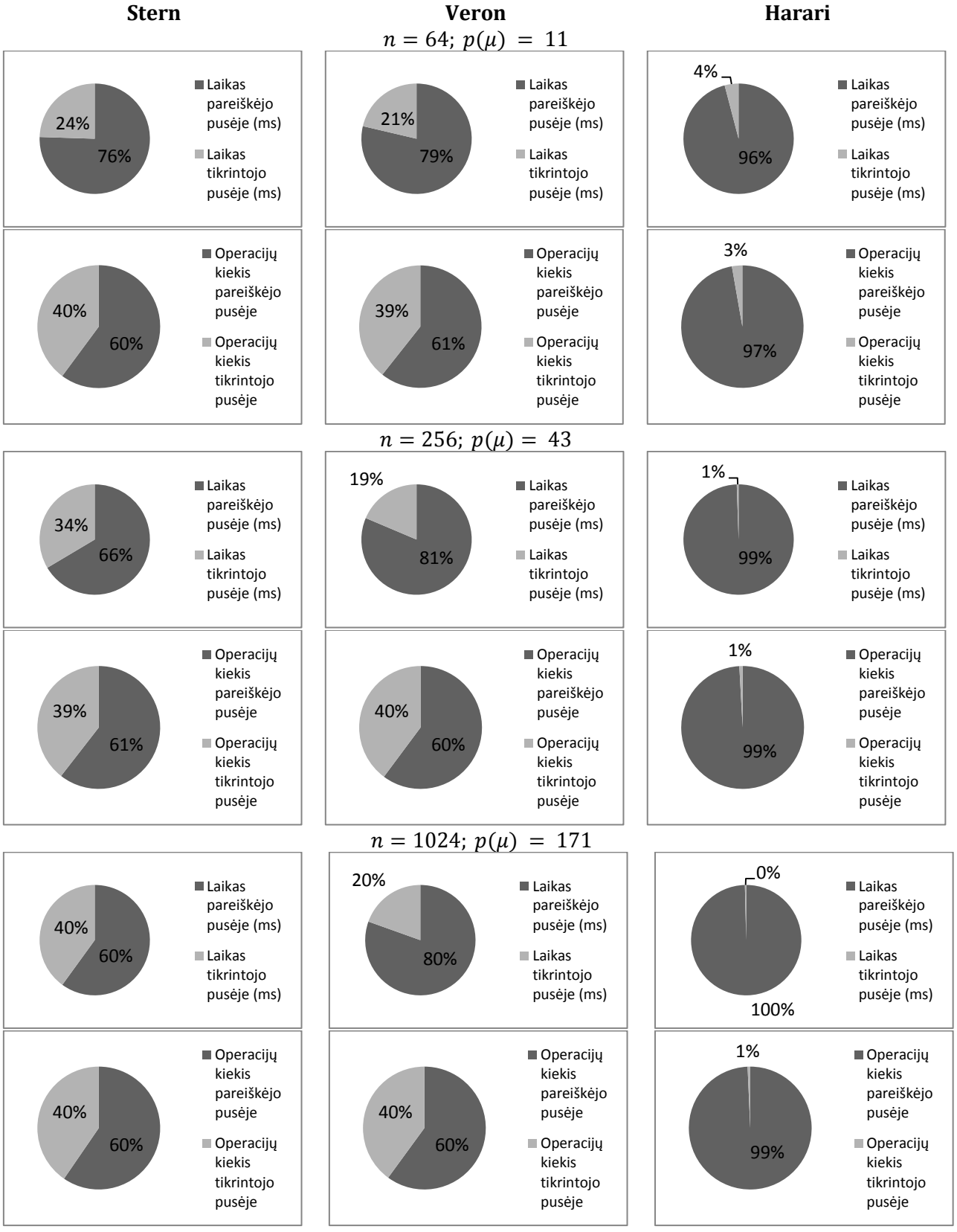


$n = 256;\ p(\mu) = 43$



$n = 1024;\ p(\mu) = 171$

3.1.4.2. Darbo pasiskirstymas



Iš gautų duomenų matome, kad didžioji darbo dalis yra atliekama pareiškėjo pusėje.

Darbo laiko pasiskirstymas ganėtinai akivaizdžiai kinta priklausomai nuo žodžių ilgio Stern ir Harari schemose. Stern schemoje matome, kad darbo laiko pasiskirstymas artėja prie lygaus žodžių ilgiui didėjant, Harari schemoje priešingai, pasiskirstymas artėja prie visiškai vienpusio.

Operacijų kiekio pasiskirstymas yra beveik nepriklausomas nuo naudojamo žodžių ilgio visose schemose. Pasiskirstymas yra pranašesnis Stern ir Veron schemose, nes didesnė darbo dalis yra atliekama tikrintojo pusėje.

3.1.4.3. Siunčiami duomenys

Ištyrę siunčiamų duomenų kiekius, galime pastebėti, kad Stern ir Veron schemų siunčiami duomenų kiekiai sutampa. Tai yra dėl to, kad abi schemos naudoja tokį patį trijų komunikacijų metodą. Kitokį, paralelinių skaičiavimų, metodą naudojanti Harari schema siunčia žymiai didesnį duomenų kiekį.

3.2. Pilnos identifikacijos tyrimas

Šio tyrimo metu pradinių kintamųjų reikšmės buvo pasirinktos pagal nuorodas pateiktas Stern schemos apraše. Šie duomenys pasirinkti, tam kad atlikti pilną identifikaciją su vienodais parametrais. Vykdymo metu, žinant teorinius atskirų iteracijų saugumus, buvo pasirinktas iteracijų kiekis, kuris užtikrintų patenkinamą saugumo lygį 10^{-6}..

Vykdymui parinkti parametrai: duomenų ilgis schemoje $N = 512$, paslapties svoris $p = 85$.

3.2.1. Stern schema

Stern schemos iteracijos teorinis saugumo lygmuo – $\frac{2}{3}$, patartinas iteracijų kiekis $r = 35$.

Tyrimas	Rezultatas
Minimalus vykdymo laikas	72,462939ms
Maksimalus vykdymo laikas	81,221726ms
Vidutinis vykdymo laikas	77,487022ms
Vidutinis laikas pareiškėjo pusėje	51,630646ms
Vidutinis laikas tikrintojo pusėje	25,856377ms
Vidutinis pareiškėjo operacijų kiekis	661881044,6
Vidutinis tikrintojo operacijų kiekis	422625950,2
Vidutinis siunčiamų duomenų kiekis	143 920B

3.2.2. Veron schema

Veron schemos iteracijos teorinis saugumo lygmuo – $\frac{2}{3}$, patartinas iteracijų kiekis $r = 35$.

Tyrimas	Rezultatas
Minimalus vykdymo laikas	42,193176ms
Maksimalus vykdymo laikas	45,383906ms
Vidutinis vykdymo laikas	43,551410ms
Vidutinis laikas pareiškėjo pusėje	34,305619ms
Vidutinis laikas tikrintojo pusėje	9,245791ms
Vidutinis pareiškėjo operacijų kiekis	332052198,8
Vidutinis tikrintojo operacijų kiekis	220602099,4
Vidutinis siunčiamų duomenų kiekis	143 920B

3.2.3. Harari schema

Harari schemos iteracijos teorinis saugumo lygmuo $-\frac{1}{2}$, patartinas iteracijų kiekis $r = 20$.

Tyrimas	Rezultatas
Minimalus vykdymo laikas	4752,001839ms
Maksimalus vykdymo laikas	4943,968371ms
Vidutinis vykdymo laikas	4820,162336ms
Vidutinis laikas pareiškėjo pusėje	4792,619721ms
Vidutinis laikas tikrintojo pusėje	27,542614ms
Vidutinis pareiškėjo operacijų kiekis	4636591298,6
Vidutinis tikrintojo operacijų kiekis	36335513,2
Vidutinis siunčiamų duomenų kiekis	6 232 960B

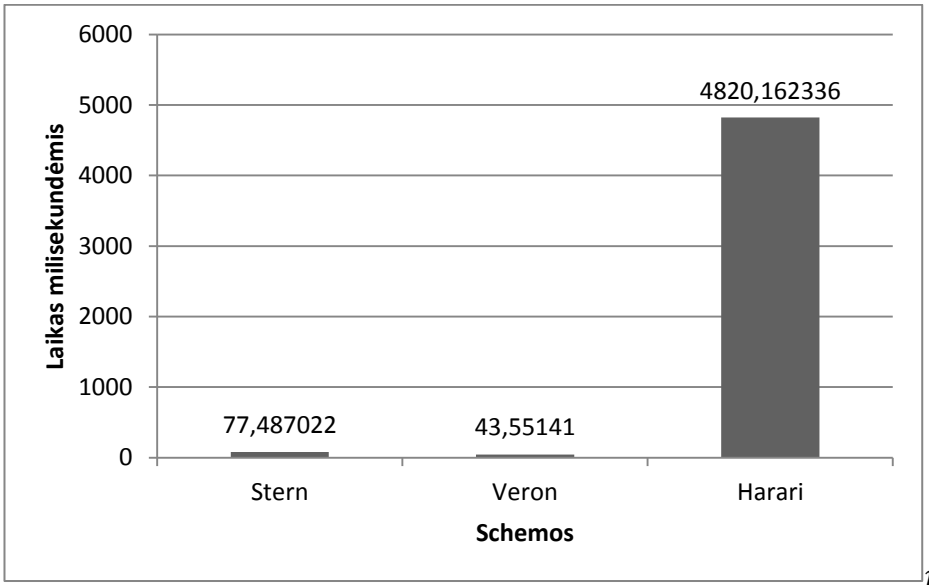
3.2.4. Palyginimas

Pilnos identifikacijos, kaip ir atskiros schemų iteracijos buvo palygintos pagal vidutinius vykdymo laikus, darbo pasiskirstymą tarp pareiškėjo ir tikrintojo, bei pagal siunčiamus duomenis.

3.2.4.1. Laiko palyginimas

Pagal tyrimo rezultatus, net kai Harari schemos iteracijų kiekis yra žymiai mažesnis (20 iteracijų, kai kitos schemos vykdo po 35), Harari schemos identifikacija užtrunka gerokai ilgiau.

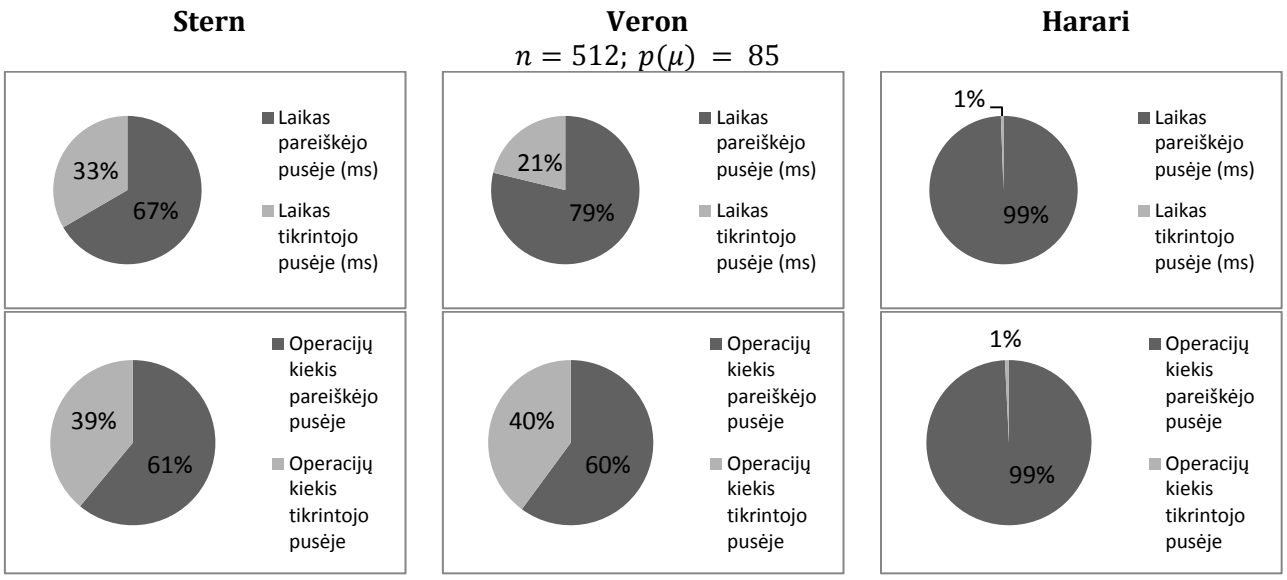
Veron schemos identifikacija, kaip ir tikėtasi, įvykdoma greičiausia.



$n = 512;\ p(\mu) = 85$

3.2.4.2. Darbo pasiskirstymas

Pilnos identifikacijos darbo pasiskirstymas proporcingai panašus atskirų iteracijų darbo pasiskirstymui. Galime matyti tas pačias tendencijas laiko bei operacijų pasidalijime.



3.2.4.3. Siunčiami duomenys

Siunčiamų duomenų tyrime taip pat matome išlikusias atskirų iteracijų tendencijas. Stern ir Veron schemos siunčia ekvivalenčius duomenų kiekius, Harari schema, net atlikdama mažiau iteracijų siunčia didesnį duomenų kiekį.

4. Rezultatai ir išvados

Darbo metu susipažinta su klaidas taisančiais kodais bei informacijos neatskleidžiančiomis identifikacijos schemomis. Parašytos programinės interpretacijos trims schemoms – Stern, Veron ir Harari.

Šios schemos ištirtos atkreipiant dėmesį į vykdymo laikus, darbų pasiskirstymą tarp pareiškėjo ir tikrintojo, bei siunčiamų duomenų kiekį. Peržiūrėtos tendencijos vykdant atskiras iteracijas bei pilnus identifikavimus.

Tyrimų rezultatai rodo, kad trimis komunikacijomis remtos schemos yra pranašesnės prieš paralelinių skaičiavimų schemą. Taip pat matome, kad Veron schema išties patobulina Stern identifikavimo modelį, tiek vykdymo laiku, tiek operacijų kiekiu.

Literatūros sąrašas

[MOV96] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press, October 1996, 816 pages

[Ske05] Gintaras Skersys. Klaidas taisančių kodų teorija. Paskaitų konspektai, 2005.

[Sta02a] Vilius Stakėnas. Kodavimo teorija. Paskaitų kursas, 2002.

[Ste94] Jacques Stern. A New Identification Scheme Based on Syndrome Decoding. In: Lecture Notes in Computer Science 773, Advances in Cryptology - CRYPTO ,93, p. 13-21, Springer-Verlag, 1994. URL: http://www.di.ens.fr/~stern/data/St47.pdf 671 Kb

[MM12] Behzad Malek, Ali Miri, Securing Harari‘s Authentication Scheme, International Journal of Network Security, Vol. 14, No. 4, PP.206-210, July 2012

[Ver97] Pascal V´eron. Improved Identification Schemes Based on Error-Correcting Codes. Applicable Algebra in Engineering, Communication and Computing, Springer Verlag, 1997